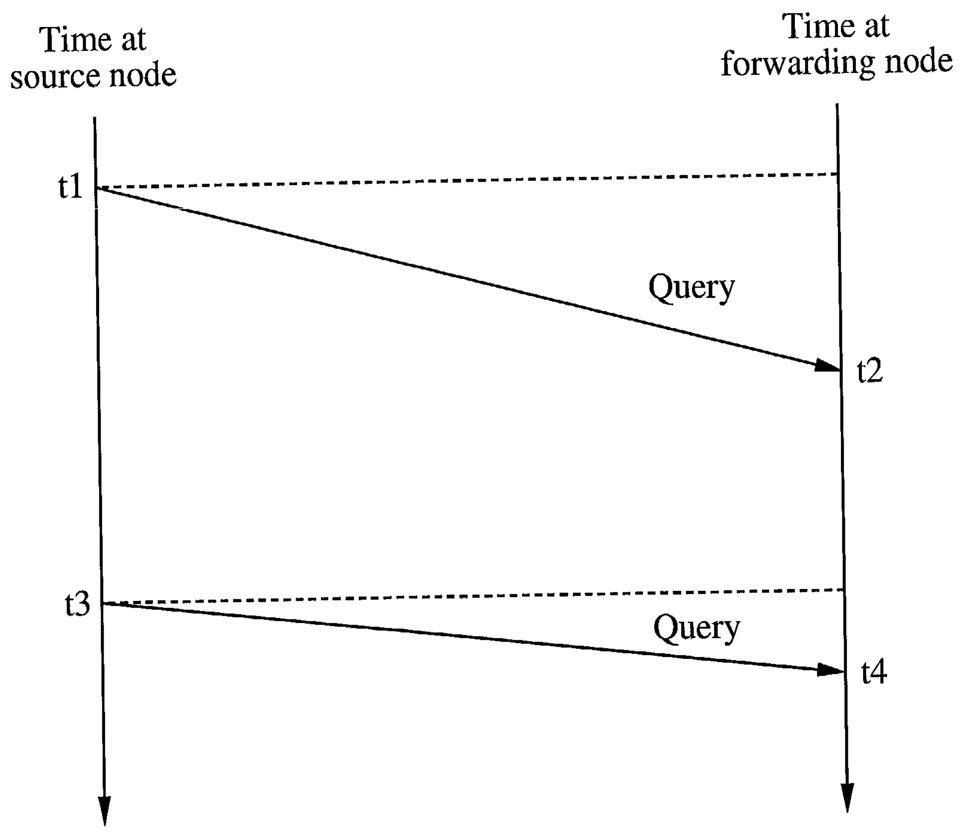

Time at
source node
Time at
forwarding node
t1
t2
t3
t4
Query
Query

FIG. 1

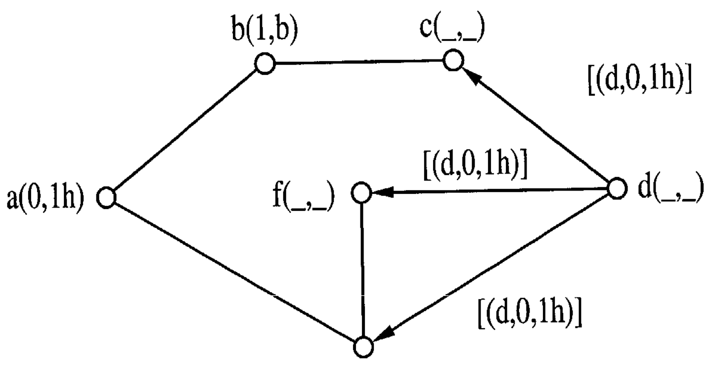


FIG. 2A

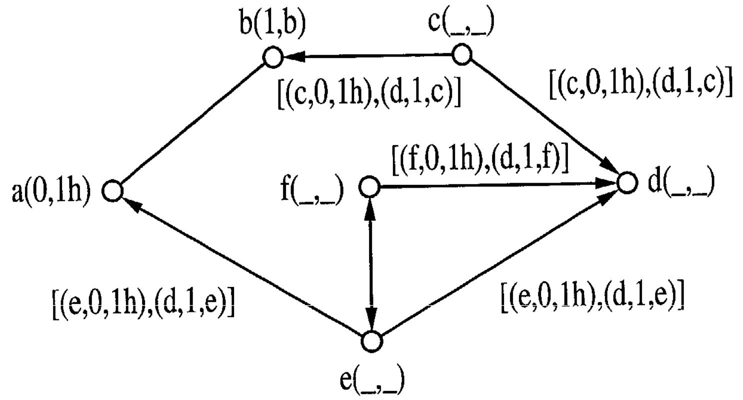


FIG. 2B

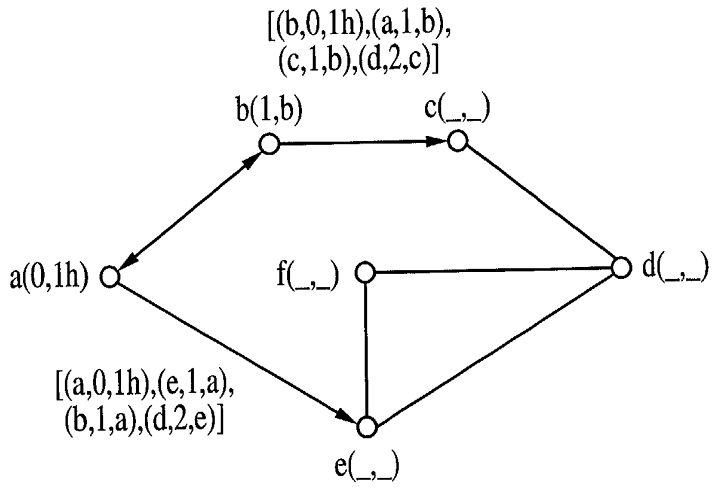


FIG. 2C

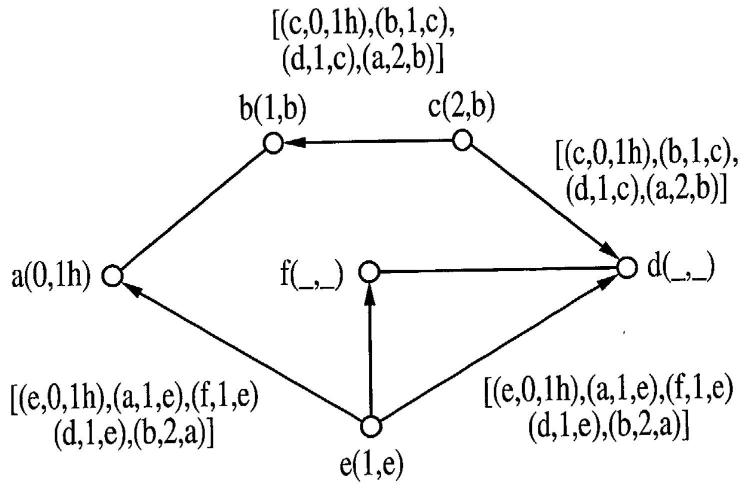


FIG. 2D

```
Procedure Init
called when node $i$ initializes itself
begin
    $N \leftarrow i$
    $D_i^i \leftarrow O$
    $s_i^i \leftarrow i$
    $p_i^i \leftarrow IP\_LOCALHOST$
    $tag_i^i \leftarrow correct$
    $T_i^i \leftarrow present\ time$
end

Procedure Recv_CU_Packet($pkt, nbr$)
when node $i$ receives a control packet from $nbr$
begin
    if ($pkt.type = QRY$)
        Query($pkt.nbr$)
    else
        if ($pkt.dst = BDCAST\_ADDR$)
            Update ($pkt,nbr$)
        else
            if($pkt.dst \in N$ and $tag_{pkt.dst}^i = correct$)
                Update ($pkt.nbr$)
    end else
end

Procedure Add_Dest($j$)
called when node $i$ learns of new destination $j$
begin
    $N \leftarrow N \cup j$
    $D_j^i \leftarrow \infty$
    $s_j^i \leftarrow NULL\_ADDR$
    $p_j^i \leftarrow NULL\_ADDR$
    $T_j^i \leftarrow$ present time
    for all ($k \in N_i$)
        $D_{jk}^i \leftarrow \infty$
        $p_{jk}^i \leftarrow NULL\_ADDR$
    end for all
end
```

FIG. 3

```
Procedure Rmv_Dest($j$)
called when node $i$ removes $j$
begin
    $N \leftarrow N - j$
    for all ($k \in N_i$)
        remove $j$ from $k$'s array
    end for all
end

Procedure Add_Nbr($k$)
called when node $i$ learns of new neighbor $k$
begin
    $N_i \leftarrow N_i \cup k$
    for all ($j \in N$)
        $D^i_{jk} \leftarrow \infty$
        $p^i_{jk} \leftarrow NULL\_ADDR$
    end for all
end

Procedure Rmv_Nbr($k$)
called when node $i$ learns of loss of neighbor $k$
begin
    $N_i \leftarrow N_i - k$
    for all ($j \in N$)
        $tag^i_j \leftarrow null$
    $send \leftarrow FALSE$
    RT_update($send$)
    If ($send = TRUE$)
        Send_Update($i$, $BDCAST\_ADDR$)
end
```

FIG. 4

```
Procedure DT_Update($k,j,RD^i_j,rp^i_j$)
updating distance table entry
begin
    if ($RD^i_j < \infty$)
        $D^i_{jk} \leftarrow RD^i_j + 1$
    else $D^i_{jk} \leftarrow \infty$
    $p^i_{jk} \leftarrow rp^i_j$
    for all ($b \in N_i$)
        if $k$ is in path from $i$ to $j$ via $b$
            $D^i_{jb} \leftarrow D^i_{kb} + RD^i_j$
    end for all
end
```

FIG. 5

```
Procedure Query($pkt,nbr$)
called for processing query
begin
  for each entry ($j$, $RD_j^i$, $rp_j^i$) in $pkt$
    if ($j \notin N$)
      if ($RD_j^i = \infty$)
        continue
      else
        Add_Dest($j$)
        if ($RD_j^i = 0$)
          Add_Nbr($j$)
      end else
    end if
    else
      if ($RD_j^i = 0$ and $j \notin N_i$)
        Add_Nbr($j$)
    end else
    DT_Update($pkt, src, j, RD_j^i, rp_j^i$)
  end for each
  $send \leftarrow FALSE$
  RT_Update($send$)
  if ($tag_{pkt.dst}^i = correct$)
    Send_Update($pkt.dst, pkt.src$)
  else
    if (present time - $qr_{pkt.dst}^i > query\_recieve\_timeout$)
      if ($pkt.hops > 1$)
        Send_Query ($pkt.dst$, ($pkt.hops - 1$), $pkt.src$)
      if ($pkt.hops \geq 1$)
        $qr_{pkt.dst}^i \leftarrow$ present time
    end if
  end else
end
```

FIG. 6

```
Procedure Update($pkt,nbr$)
called for processing update
begin
   $newpath \leftarrow FALSE$
   if ($pkt.dst \neq BDCAST\_ADDR$)
      if ($pkt.src \notin N$ or $tag^i_{pkt.src} \neq correct$)
         $newpath \leftarrow TRUE$
   for each entry ($j$, $RD^i_j$, $rp^i_j$) in $pkt$
      if ($j \notin N$)
         if ($RD^i_j = \infty$)
            continue
         else
            Add_Dest($j$)
            if ($RD^i_j = 0$)
               Add_Nbr($j$)
         end else
      end if
   else
      if ($RD^i_j = 0$ and $j \notin N_i$)
         Add_Nbr($j$)
   end else
   DT_Update(pkt.src, j, $RD^i_j$, $rp^i_j$)
 end for each
 $send \leftarrow FALSE$
 RT_Update($send$)
 if ($pkt.dst = BDCAST\_ADDR$)
    if ($send = TRUE$) then Send_Update($i$, $BDCAST\_ADDR$)
 else
    if ($pkt.dst = i$)
       if ($send = TRUE$) then Send_Update($i$, $BDCAST\_ADDR$)
    else
       if ($newpath = TRUE$ and ($pkt.src \in N$ or $tag^i_{pkt.src} \neq correct$))
          $newpath \leftarrow FALSE$
       if ($tag^i_{pkt.dst} = correct$ and $newpath = TRUE$
          and $pkt.src$ is not in the path to $pkt.dst$)
          Send_Update($pkt.src$, $pkt.dst$)
    else
       if ($send$) then Send_Update($i$, $BDCAST\_ADDR$)
  end else
end else
end
```

FIG. 7

```
Procedure RT_Update($send$)
updating routing table entries
begin
  for all ($j \in N$)
    if ($j = i$)
      continue
    $DTMin \leftarrow Min\ \{D^i_{jb}\ \forall b \in N_i\}$
    if ($D^i_{js^i_j} = DTMin$) then $ns \leftarrow s^i_j$
    else $ns \leftarrow b | \{b \in N_i$ and $D^i_{jb} = DTMin\}$
    $x \leftarrow j$
    $loop \leftarrow FALSE$
    for ($m = 0$; $m < |N|$; $m$ ++)
      $visited[m] \leftarrow NULL\_ADDR$
    $num\_visited \leftarrow 0$
    while (($D^i_{xns} = Min\ \{ D^i_{xb}\ \forall b \in N_i \}$)
        and $D^i_{xns} < \infty$ and $tag^i_x \leftarrow null$ and $loop = FALSE$)
        $m \leftarrow 0$
        while ($m < num\_visited$)
          if ($visited\ [m] = x$ or $x = i$)
            $loop \leftarrow TRUE$
        end while
        $x \leftarrow p^i_{xns}$
    end while
    if ($loop = FALSE$ and ($p^i_{xns} = IP\_LOCALHOST$ or $tag^i_x = correct$))
        $tag^i_j \leftarrow correct$
    else
        $tag^i_j \leftarrow error$
    if ($tag^i_j = correct$)
      if ($D^i_j < DTMin$) then $send \leftarrow TRUE$
      $D^i_j \leftarrow DTMin$
      $s^i_j \leftarrow ns$
      if ($D^i_j = 1$) then $p^i_j \leftarrow i$
      else $p^i_j \leftarrow p^i_j$
    end if
    end
      if ($D^i_j = \infty$) then $send \leftarrow TRUE$
      $p^i_j \leftarrow NULL\_ADDR$
      $s^i_j \leftarrow NULL\_ADDR$
      $D^i_j \leftarrow \infty$
     end else
  end for all
end
```

FIG. 8

```
Procedure Send_Update($src, dst$)
broadcasting update
begin
    for each entry e ($j$ , $D^i_j$, $p^i_j$) in routing table
        $LIST \leftarrow LIST + e$
    sort all entries in $LIST$ in ascending distance values
    add each entry in $LIST$ to $pkt$
    $pkt.dst \leftarrow dst$
    $pkt.src \leftarrow src$
    $pkt.type \leftarrow UPDATE$
    broadcast $pkt$ to all neighbors
end

Procedure Send_Query($dest, hops, src$)
broadcasting query
begin
    for each entry $e$ ($j$, $D^i_j$, $p^i_j$) in routing table
        $LIST \leftarrow LIST + e$
    sort all entries in $LIST$ in ascending distance values
    add each entry in $LIST$ to $pkt$
    $pkt.dst \leftarrow dest$
    $pkt.src \leftarrow src$
    $pkt.hops \leftarrow hops$
    $pkt.type \leftarrow QUERY$
    broadcast $pkt$ to all neighbors
end

Procedure Buffer_Timer_Callback()
called periodically when buffer timer expires begin
    $send \leftarrow FALSE$
    Check_Buffer($send$)
    if (send = TRUE) then Send_Update($i$, BDCAST_ADDR)
end
```

FIG. 9

```
Procedure Get_Route_For_Pkt($dest$)
decides if query needs to be sent and sends it
begin
   If (((present time - $qs^{i}_{dest}$) > $query\_send\_timeout$)
        and ($hqs^{i}_{j} = MAX\_HOPS$ ))
        $hqs^{i}_{j} \leftarrow ZERO$
        $zqs^{i}_{j} \leftarrow$ present time
        $Send\_Query(dest, 0, i)$
end if
If (((present time - $qs^{i}_{dest}$) > $query\_send\_timeout$)
        and ($hqs^{i}_{j} = ZERO$ )
        and (present time - $zqs^{i}_{j}$) > zero_$qry\_send\_timeout$)
        $hqs^{i}_{j} \leftarrow MAX\_HOPS$
        $qs^{i}_{j} \leftarrow$ present time
        $qr^{i}_{j} \leftarrow$ present time
        Send_Query($dest, MAX\_HOPS, i$)
   end if
end if
```

FIG. 10

```
Procedure Handle_Data_Pkt ($pkt,nbr$)
data packet can be from an upper layer or a forwarded pkt from $nbr$
begin
    If ( $pkt.dst$ = i)
        send packet to correct upper layer port
    else if ( $pkt.src$ = i)
        $j \leftarrow pkt.dst$
        if ($j \in N$ and $tag_j^i = correct$)
            send $pkt$ to $s_j^i$
        else
            queue $pkt$ in buffer
            Get_Route_For_Pkt ( $pkt.dst$)
        end else
      end else if
      else
        $j \leftarrow pkt.dst$
        if( $j \in N$)
        if ($nbr = s_j^i$)
          Send_Update($i$, $BDCAST\_ADDR$)
          drop $pkt$ and return
        end if
        if ($tag_j^i = correct$)
          send $pkt$ to $s_j^i$
      else
          Send_Update($i$, $BDCAST\_ADDR$)
          drop $pkt$
      end if
      else
        queue $pkt$ in buffer
        Get_Route_For_Pkt ( $pkt.dst$)
      end else
    end else
  end
```

FIG. 11

```
Procedure Check_Buffer($send$)
checks buffer to forward or drop packets
begin
    for each  $pkt$ in buffer
      if (time $pkt$ has been in buffer > $data\_pkt\_timeout$)
          drop $pkt$
          return
      end if
      $j \leftarrow pkt.dst$
      if ($pkt.src = i$)
          if ($j \in N$ and $tag_j^i = correct$)
              send $pkt$ to $s_j^i$
      else
              Get_Route_For_Pkt($pkt.dst$)
      end if
      else
      if ($j \in N$)
          if($tag_j^i = correct$)
              send $pkt$ to $s_j^i$
          else
              $send \leftarrow TRUE$
              drop $pkt$
            end else
           end if
         else
              Get_Route_For_Pkt($pkt.dst$)
       end else
    end for each
end
```

FIG. 12

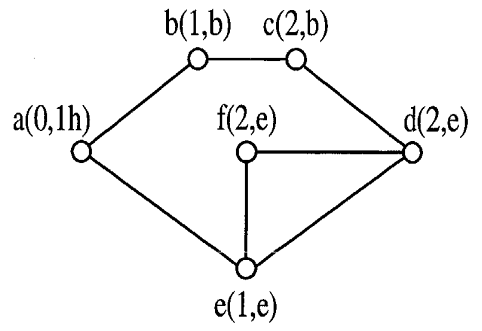


FIG. 13A

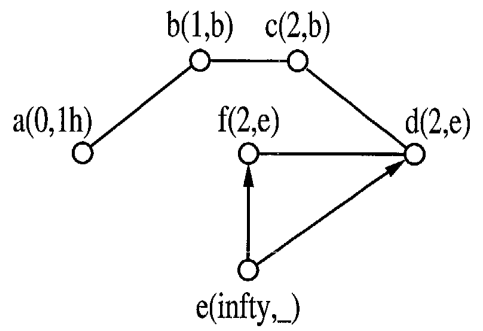


FIG. 13B

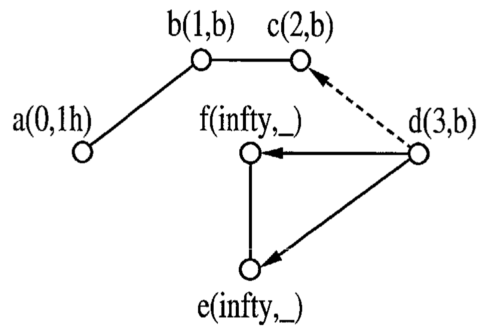


FIG. 13C

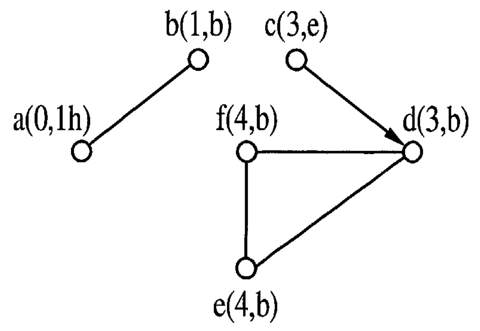


FIG. 13D

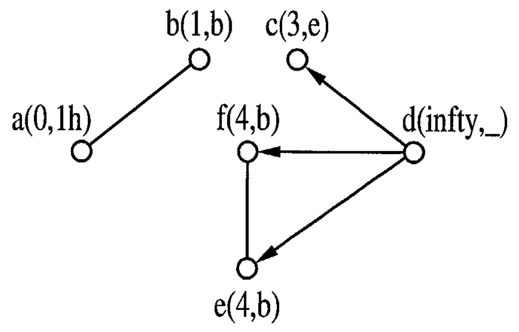


FIG. 13E

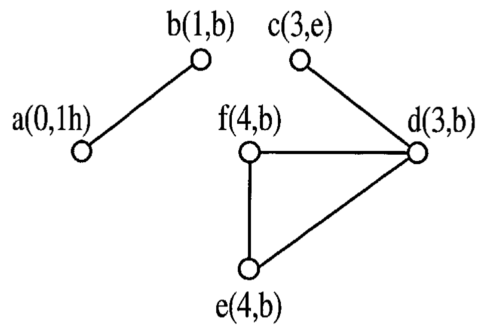
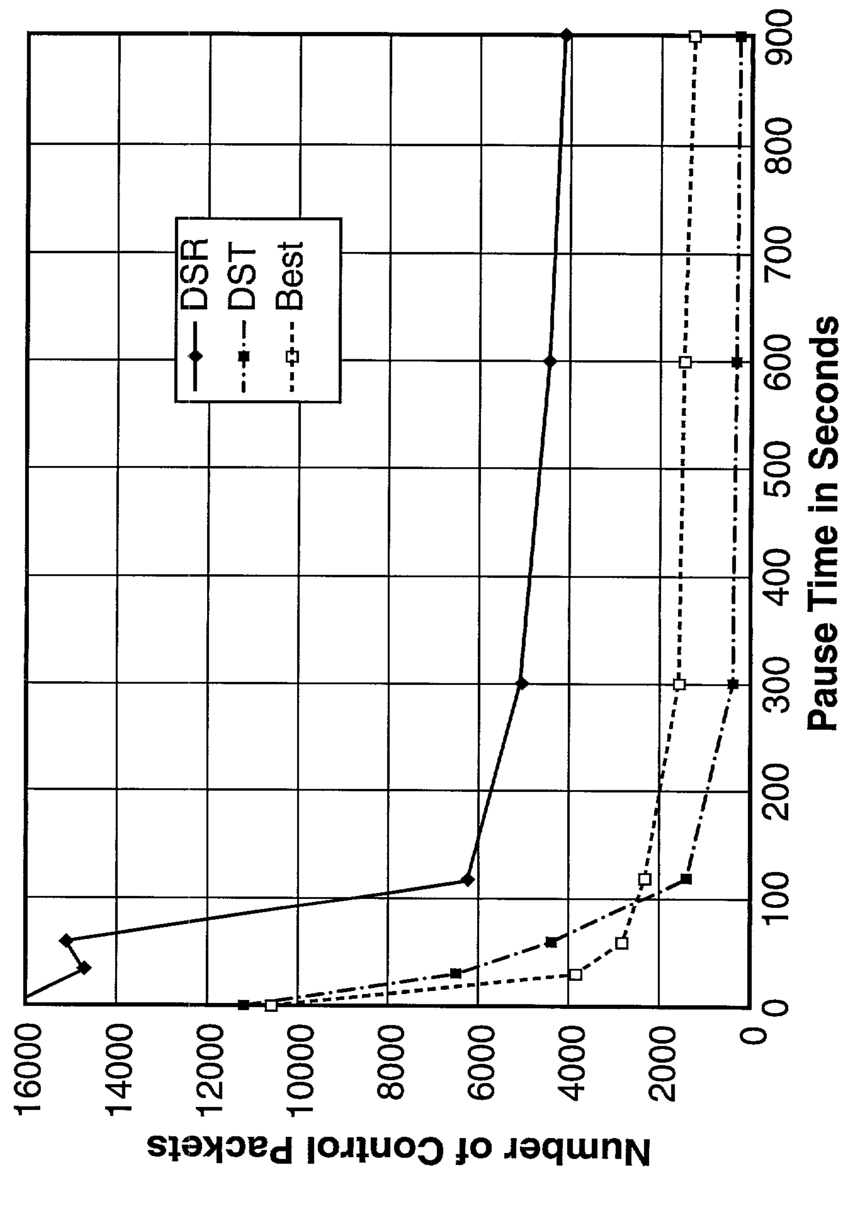


FIG. 13F



Number of Control Packets
16000
14000
12000
10000
8000
6000
4000
2000
0
0
100
200
300
400
500
600
700
800
900
Pause Time in Seconds
DSR
DST
Best

FIG. 14

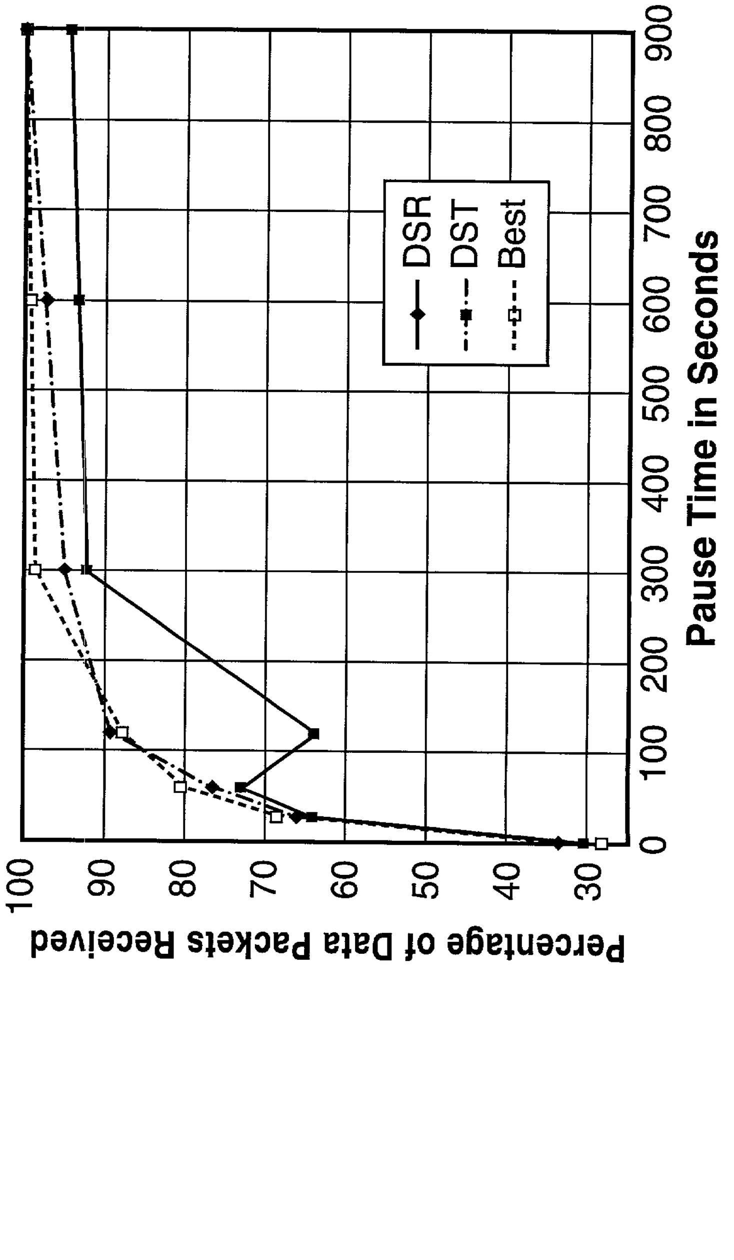
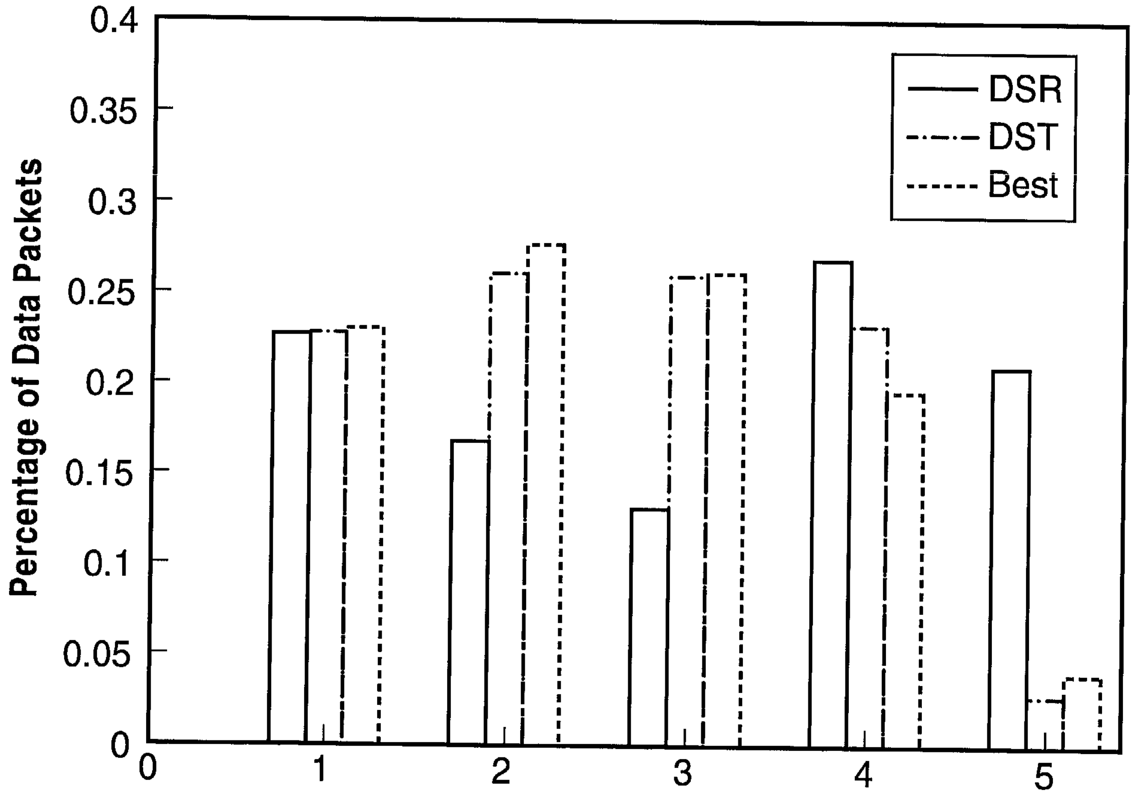

Percentage of Data Packets Received
100
90
80
70
60
50
40
30
0
100
200
300
400
500
600
700
800
900
Pause Time in Seconds
DSR
DST
Best

FIG. 15



Number of Hops

FIG. 16

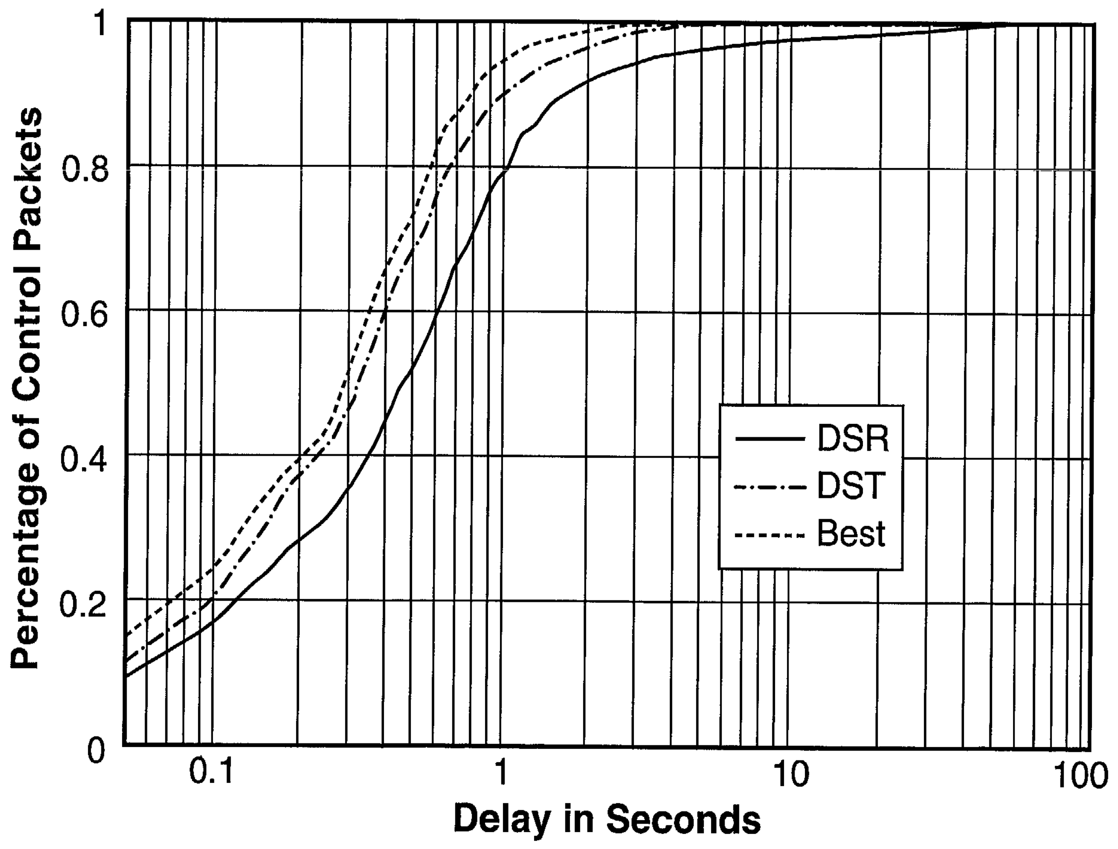

1
0.8
0.6
0.4
0.2
0
Percentage of Control Packets
0.1
1
10
100
Delay in Seconds
DSR
DST
Best

FIG. 17

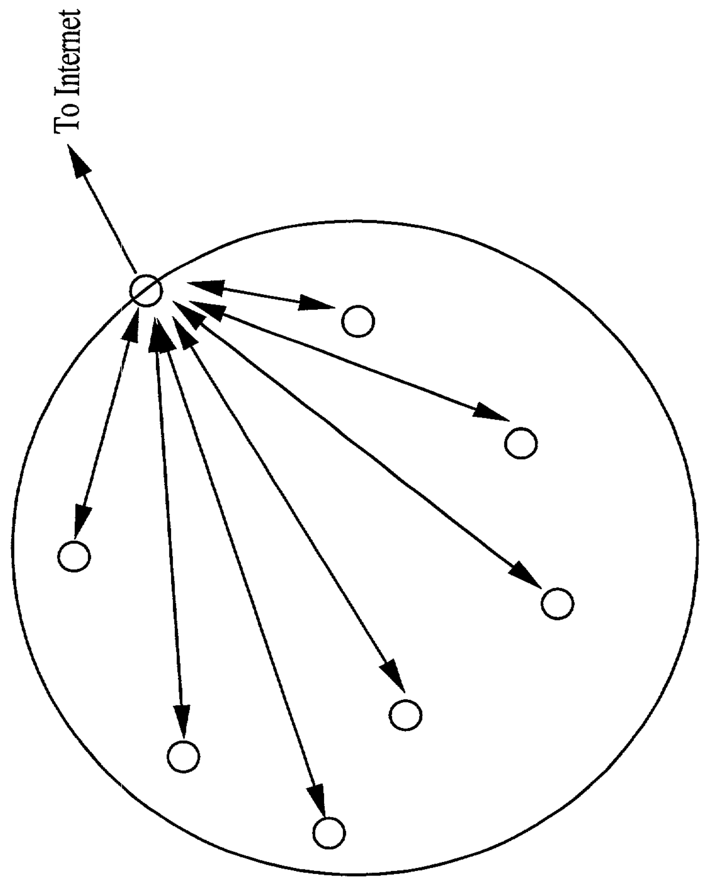


FIG. 18

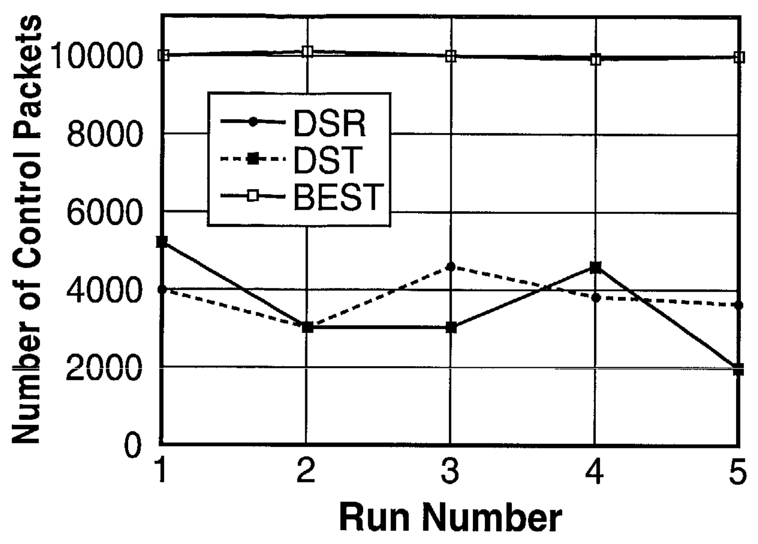


FIG. 19

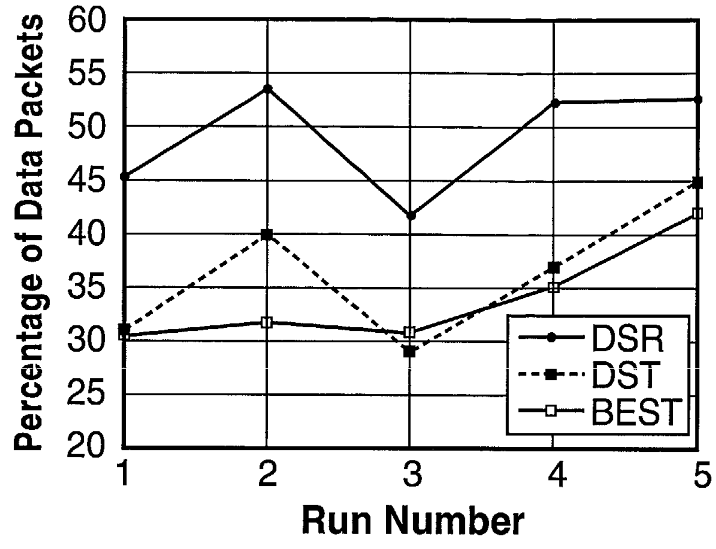


FIG. 20

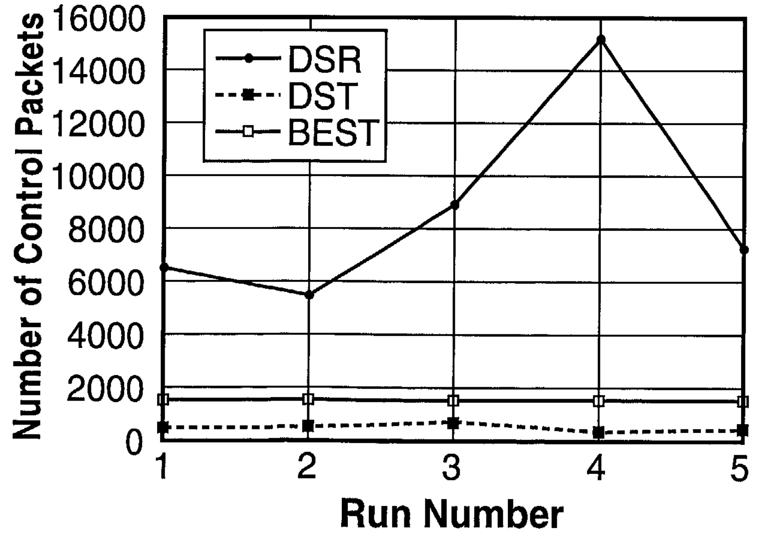


FIG. 21

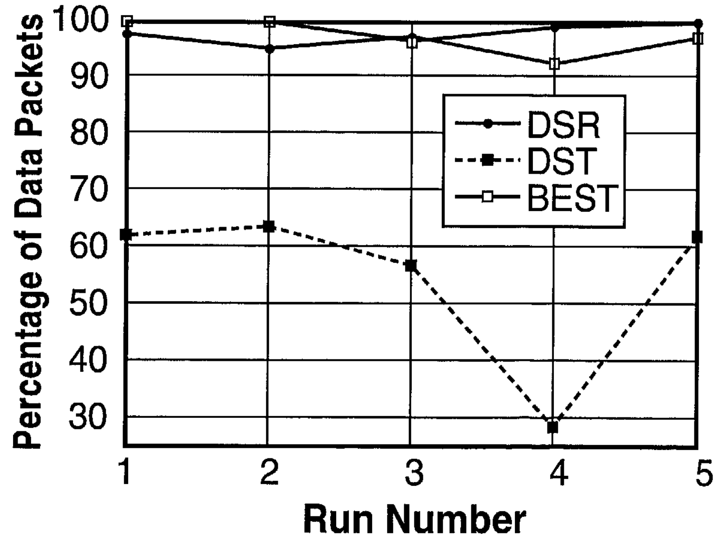


FIG. 22